GRU AND LSTM CODE

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, GRU, Dense

np.random.seed(0)
tf.random.set_seed(0)

num_words = 10000
max_sequence_length = 100

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words  = num_words)
x_train = pad_sequences(x_train, maxlen = max_sequence_length)
x_test = pad_sequences(x_test, maxlen = max_sequence_length)

model = Sequential()

model.add(Embedding(input_dim = num_words, output_dim = 32, input_lenght = max_sequence_length)
model.add(GRU(units = 32))
model.add(Dense(units = 1, activation = ‘sigmoid’)

model.compile(optimizer = ‘adam’, loss=’binary_crossentropy’, metrics = [‘accuracy’])

model.fit(x_train , batch_size = 64, epochs = 7, validation_split = 0.2)

loss, accuracy = model.evaluate(x_test, y_test)
print(f”Test loss: {loss:.4f}, Test accuracy: {accuracy:.4f}”)
```

LSTM CODE:

```
import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, LSTM, Dense

import matplotlib.pyplot as plt


def generate_time_series_data(num_points):

        t = np.linspace(0, 10, num_points)

        data = np.sin(t) + 0.1 * np.random.randn(num_points)

        return data


num_points = 1000

data = generate_time_series_data(num_points)


sequence_points = 10

X = []

Y = []

for i in range (num_points – sequence_length):

        X.append(data[i: i+sequence_length])

        y.append(data[i+sequence_length])


X = np.array(X).reshape(-1, sequence_length, 1)

y = np.array(y)


train_ratio = 0.8

train_size = int(train_ratio * len(X))

X_train, X_test = X[: train_size], X[train_size:]

y_train, y_test = y[: train_size], y[train_size:]


model = Sequential()
```

```
model.add(LSTM(units = 50, input_size = (sequence_length, 1)))
model.add(Dense(units = 1))
model.compile(optimizer = ‘adam’, loss = ‘mean_squared_error’)
history = model.fit(X_train, y_train, batch_size = 32, epochs = 15, validation_split = 0.2)
loss = model.evaluate(X_test, y_test)
print(f”test loss: {loss:.4f}”)
predictions = model.predict(X_test)
plt.figure(figsize = (12, 6))
plt.plot(y_test, label = ‘Actual’)
plt.plot(predictions, label = ‘Predicted’)
plt.legend()
plt.title(‘actual and predicted’)
plt.xlabel(‘time’)
plt.ylabel(‘value’)
plt.show()
```